signals HIGH.

Comparing the state of the pipeline in Cycle 4 and Cycle 5, it can be seen that the provision of secondary storage elements, enables the pipeline embodiment shown in Fig. 2 to expand, that is, to free up data storage elements into which valid data can be advanced. For example, in Cycle 4, the data blocks D1, D2 and D3 form a "solid wall" since their data cannot be transferred until the ACCEPT signal into Stage F goes HIGH. Once this signal does become HIGH, however, data D1 is shifted out of the pipeline, data D2 is shifted into the primary storage elements of Stage F, and the secondary storage elements of Stage F become free to accept new data if the following device is not able to receive the data D2 and the pipeline must once again "compress". This is shown in Cycle 6, for which the data D3 has been shifted into the secondary storage elements of Stage F and the data D4 has been passed on from Stage D to Stage E as normal.

Figs. 3a(1), 3a(2), 3b(1) and 3b(2) (which are referred to collectively as Fig. 3) illustrate generally a preferred embodiment of the pipeline. This preferred embodiment implements the structure s hown in F igure 2 u sing a two-phase, non-overlapping clock with phases ø0 and ø1. Although a two-phase clock is preferred, it will be appreciated that it is also possible to drive the various embodiments of the invention using a clock with more than two phases.

As shown in Fig. 3, each pipeline stage is represented as having two separate boxes which illustrate the primary and secondary storage elements. Also, although the VALID signal and the data lines connect the various pipeline stages as before, for ease of illustration, only the ACCEPT signal is shown in Fig. 3. A change of state during a clock phase of certain of the ACCEPT signals is indicated in Fig. 3 using an upward-pointing arrow for changes from LOW to HIGH. Similarly, a downward-pointing arrow for changes from HIGH to

LOW. Transfer of data from one storage element to another is indicated by a large open arrow. It is assumed that the VALID signal out of the primary or secondary storage elements of any given stage is HIGH whenever the storage elements contain valid data.

In Fig. 3, each cycle is shown as consisting of a full period of the non-overlapping clock phases ø0 and ø1. As is explained in greater detail below, data is transferred from the secondary storage elements (shown as the left box in each stage) to the primary storage elements (shown as the right box in each stage) during clock cycle ø1, whereas data is transferred from the primary storage elements of one stage to the secondary storage elements of the following stage during the clock cycle ø0. Figure 3 also illustrates that the primary and secondary storage elements in each stage are further connected via an internal acceptance line to pass an ACCEPT signal in the same manner that the ACCEPT signal is passed from stage to stage. In this way, the secondary storage e lement will know when it can pass its date to the primary storage element.

Fig. 3 shows the ø1 phase of Cycle 1, in which data D1, D2 and D3, which were previously shifted into the secondary storage elements of Stages E, D, and B, respectively, are shifted into the primary storage elements of the respective s tage. D uring the ø 1 p hase o f C ycle 1, t he p ipeline, t herefore, assumes the same configuration as is shown as Cycle 1 of FIG 2. As before, the ACCEPT signal into Stage F is assumed to be LOW. As Fig. 3 illustrates, however, this means that the ACCEPT signal into the primary storage element of Stage F is LOW, but
since this storage element does not contain valid data, it sets the ACCEPT signal into its secondary storage element HIGH.

The ACCEPT signal from the secondary storage elements of Stage F into the primary storage elements of Stage E is also set HIGH since the secondary storage elements of Stage F do not contain valid data. As before, since the primary storage elements of Stage F are able to accept data, data in all the upstream primary and secondary storage elements can be shifted downstream without any valid data being overwritten. The shift of data from one stage to the next takes place during the next ø0 phase in Cycle 2. For example, the valid data D1 contained in the primary storage element of Stage E is shifted into the secondary storage element of Stage F, the data D4 is shifted into the pipeline, that is, into the secondary storage element of Stage A, and so forth.

The primary storage element of Stage F still does not contain valid data during the ø0 phase in Cycle 2 and, therefore, the ACCEPT signal from the primary storage elements into the secondary storage elements of Stage F remains HIGH. During the ø0 phase in Cycle 2, data can therefore be shifted yet another step to the right, i.e., from the secondary to the primary storage elements within each stage.

However, once valid data is loaded into the primary storage elements of Stage F, if the ACCEPT into Stage F from the downstream device is still LOW, it is not possible to shift data out of the secondary storage element of Stage F without overwriting and destroying the valid data D1. The ACCEPT signal from the primary storage elements into the secondary storage elements of Stage F therefore goes LOW. Data D2, however, can still be shifted into the secondary storage of Stage F since it did not contain valid data and its ACCEPT signal out was HIGH.

During the ø1 phase of Cycle 3, it is not possible to shift data D2 into the primary storage elements of Stage F, although data can be shifted within all the previous stages. Once valid data is loaded into the secondary storage elements of Stage F, however, Stage F is not able to pass on this data. It signals this event setting its ACCEPT signal out LOW.

Assuming that the ACCEPT signal into Stage F remains LOW, data upstream of Stage F can continue to be shifted between stages and within stages on the respective clock phases until the next valid data block D3 reaches the primary storage elements of Stage E. As illustrated, this condition is reached during the ø1 phase of Cycle 4.

During the ø0 phase of Cycle 5, data D3 has been loaded into the primary storage element of Stage E. Since this data cannot be shifted further, the ACCEPT signal out of the primary storage elements of Stage E is set LOW. Upstream data can be shifted as normal.

Assume now, as in Cycle 5 of Fig. 2, that the device connected downstream of the pipeline is able to accept pipeline data. It signals this event by setting the ACCEPT signal into pipeline Stage F HIGH during the ø1 phase of Cycle 4. The primary storage elements of Stage F can now shift data to the right and they are also able to accept new data. Hence, the data D1 was shifted out during the ø1 phase of Cycle 5 so that the primary storage elements of Stage F no longer contain data that must be saved. During the ø1 phase of Cycle 5, the data D2 is, therefore, shifted within Stage F from the secondary storage elements to the primary storage elements. The secondary storage elements of Stage F are also able to accept new data and signal this by setting the ACCEPT signal into the primary storage elements of Stage E HIGH. During transfer of data within a stage, that is, from its secondary to its primary storage elements, both sets of storage elements will contain the same data, but the data in the secondary storage elements can be overwritten with no data loss since this data will also be held in the primary storage elements. The same holds true for data transfer from the primary storage elements of one stage into the secondary storage elements of a subsequent stage.

Assume now, that ACCEPT signal into the primary storage elements of Stage F goes LOW during the ø1 phase in Cycle 5. This means that Stage F is not able to transfer the data D2 out of the pipeline. Stage F, consequently, sets the ACCEPT signal from its primary to its secondary storage elements LOW to prevent overwriting of the valid data D2. The data D2 stored in the secondary storage elements of Stage F, however, can be overwritten without loss, and the data D3, 10 is therefore, transferred into the secondary storage elements of Stage F during the ø0 phase of Cycle 6. Data D4 and D5 can be shifted downstream as normal. Once valid data D3 is stored in Stage F along with data D2, as long as the ACCEPT signal into the primary storage elements of Stage F is LOW, neither of the secondary storage elements can accept new data, and it signals this by setting the ACCEPT signal into Stage E LOW.

When the ACCEPT signal into the pipeline from the downstream device changes from LOW to HIGH or vice versa, this change does not have to propagate upstream within the pipeline further than to the immediately preceding storage elements (within the same stage or within the preceding pipeline stage). Rather, this change propagates upstream within the pipeline one storage element block per clock phase.

As this example illustrates, the concept of a "stage" in the pipeline structure illustrated in Fig. 3 is to some extent a matter of perception. Since data is transferred within a stage (from the secondary to the primary storage elements) as it is between stages (from the primary storage elements of the upstream stage into the secondary storage elements of the neighboring downstream stage), one could just as well consider a stage to consist of "primary" storage elements followed by "secondary storage elements" instead of as illustrated in Fig. 3. The concept of "primary" and "secondary" storage elements is, therefore, mostly a question of labeling. In Fig. 3, the "primary" storage elements can also be referred to as "output" storage elements, since they are the elements from which data is transferred out of a stage into a following stage or device, and the "secondary" storage elements could be "input" storage elements for the same stage.

In explaining the forementioned embodiments, as shown in Figs. 1-3, only the transfer of data under the control of the ACCEPT and VALID signals has been mentioned. It is to be further understood that each pipeline stage may also process the data it has received arbitrarily before passing it between its internal storage elements or before passing it to the following pipeline stage. Therefore, referring once again to Fig. 3, a pipeline stage can, therefore, be defined as the portion of the pipeline that contains input and output storage elements and that arbitrarily processes data stored in its storage elements.

Furthermore, the "device" downstream from the pipeline Stage F, need not be some other type of hardware structure, but rather it can be another section of the same or part of another pipeline. As illustrated below, a pipeline stage can set its ACCEPT signal LOW not only when all of the downstream storage elements are filled with valid data, but also when a stage requires more than one clock phase to finish processing its data. This also can occur when it creates valid data in one or both of its storage elements. In other words, it is not necessary for a stage simply to pass on the ACCEPT signal based on whether or not the immediately downstream storage elements contains valid data that cannot be passed on. Rather the ACCEPT signal itself may also be altered within the stage or, by circuitry external to the stage, in order to control the passage of data between adjacent storage elements. The VALID signal may also be processed in an analogous manner.

A great advantage of the two-wire interface (one wire for each of the VALID and ACCEPT signals) is its ability to control the pipeline without the control signals needing to propagate back up the pipeline all the way to its beginning stage. Referring once again to Fig. 1, Cycle 3, for example, although stage F "tells" stage E that is cannot accept data, and stage E tells stage D, and stage D tells stage C. Indeed, it there had been more stages containing valid data, then this signal would have propagated back even further along the pipeline. In the embodiment shown in Fig. 3, Cycle 3, the LOW ACCEPT signal is not propagated any further upstream than to Stage E and, then, only to its primary storage elements.

As described below, this embodiment is able to achieve this flexibility without adding significantly to the silicon area that is required to implement the design. Typically, each latch in the pipeline used for data storage requires only a single extra transistor (which lays out very efficiently in silicon). In addition, two extra latches and a small number of gates are preferably added to process the ACCEPT and VALID signals that are associated with the data latches in each half-stage.

Fig. 4 illustrates a hardware structure that implements a stage as shown in Fig. 3.

By way of example only, it is assumed that eight-bit data is to be transferred (with or without further manipulation in optional combinatorial logic circuits) in parallel through the pipeline. However, it will be appreciated that either more or less than eight-bit data can be used in practicing the invention. Furthermore, the two-wire interface in accordance with the embodiment is, however, suitable for use with any data bus width, and the data bus width may even change from one stage to the next if a particular application so requires. The interface in accordance with this embodiment can also be used to process analog signals.

As discussed previously, while other conventional timing arrangements may be used, the interface is preferably controlled by a two-phase, non-overlapping clock. In FIGS. 4-9, these clock phase signals are referred to as PH0 and PH1. In Fig. 4, a line is shown for each clock phase signal.

Input data enters a pipeline stage over a multi-bit data bus IN_DATA and is transferred to a following pipeline stage or to subsequent receiving circuitry over an output data bus OUT_DATA. The input data is first loaded in a manner described below into a series of input latches (one of each input data signal) collectively referred to as LDIN, which constitute the secondary storage elements described above.

In the illustrated example of this embodiment, it is assumed that the Q outputs of all latches follow their D inputs, that is, they are "loaded", when the clock input is HIGH, i.e., at a logic "1" level. Additionally, the Q outputs hold their last values. In other words, the Q outputs are "latched" on the falling edge of their respective clock signals. Each latch has for its clock either one of two non-overlapping clock signals PH0 and PH1 (as shown in Fig. 5), or the logical AND combination of one of these clock signals PH0, PH1 and one logic signal. The invention works equally well, however, by providing latches that latch on the rising edges of the clock signals, or any other known latching arrangement, as long as conventional methods are applied to ensure proper timing of the latching operations.

The output data from the input data latch LDIN passes via an arbitrary and optional combinatorial logical circuit B1, which may be provided to convert output data from input latch LDIN into intermediate data, which is then later loaded in an output latch LDOUT, which comprises the primary storage elements described above. The output from the output data latch LDOUT may similarly pass through an arbitrary and optional combinatorial logic circuit B2 before being passed onward as OUT_DATA to the next device downstream. This may be another pipeline stage or any other device connected to the pipeline.

In the practice of the present invention, each stage of the pipeline also includes a validation input latch LVIN, a validation output latch LVOUT, an acceptance input latch LAIN, and an acceptance output latch LAOUT. Each of these four latches is, preferably, a simple, single-stage latch. The outputs from latches LVIN, LVOUT, LAIN and LAOUT are, respectively, QVIN, QVOUT, QAIN, QAOUT. The output signal QVIN from the validation input latch is connected either directly as an input to the validation output latch LVOUT, or via intermediate logic devices or circuits that may alter the signal.

Similarly, the output validation signal QVOUT of a given stage may be connected either directly to the input of the validation input latch QVIN of the following stage, or via intermediate devices or logic circuits, which may alter the validation signal. This output QVIN is also connected to a logic gate (to be described below), whose output is connected to the input of the acceptance input latch LAIN. The output QAOUT from the acceptance output latch LAOUT is connected to a similar logic gate (described below), optionally via another logic gate.

As shown in Fig. 4, the output validation signal QVOUT forms an OUT_VALID signal that can be received by subsequent stages as an IN_VALID signal, or simply to indicate valid data to subsequent circuitry connected to the pipeline. The readiness of the following circuit or stage to accept data is indicated to each stage as the signal OUT_ACCEPT, which is connected as the input to the acceptance output latch LAOUT, preferably via logic circuitry, which is described below. Similarly, the output QAOUT of the acceptance output latch LAOUT is connected as the input to the acceptance input latch LAIN, preferably via logic circuitry, which is described below.

In practicing the present invention, the out signals QVIN, QVOUT from the validation latches LVIN, LVOUT are combined with the acceptance signals QAOUT, OUT_ACCEPT, respectively, to form the inputs to the acceptance latches LAIN, LAOUT, respectively. In the embodiment illustrated in Fig. 4, these input signals are formed as the logical NAND combination of the respective validation signals QVIN, QVOUT, with the logical inverse of the representative acceptance output signals QAOUT, OUT_ACCEPT. Conventional logic gates, NAND1 and NAND2, perform the NAND operation, and the inverters INV1, INV2 form the logical inverses of the respective acceptance signals.

As is well known in the art of digital design, the output from a NAND gate is a logical "1" when any or all of its input signals are in the logical "0" state. The output from a NAND gate is, therefore, a logical "0" only when all of its inputs are in the logical "1" state. Also well known in the art, is that the output of a digital inverter such as INV1 is a logical "1" when its input signal is a "0" and is a "0" when its input signal is a "1"

The inputs to the NAND gate NAND1 are, therefore, QVIN and NOT (QAOUT), where "NOT" indicates binary inversion. Using known techniques, the input to the acceptance latch LAIN can be resolved as follows:

NAND(QVIN, NOT(QAOUT)) = NOT (QVIN) OR QAOUT

In other words, the combination of the inverter INV1 and the NAND gate NAND1 is a logical "1" either when the signal QVIN is a "0" or the signal QAOUT is a "1" either when the signal QVIN is a "0" or the signal QAOUT is a "1", or both. The gate NAND1 and the inverter INV1 can, therefore, be implemented by a single OR gate that has one of its inputs tied directly to the QAOUT output of the acceptance latch LAOUT and its other input tied to the inverse of the output signal QVIN of the validation input latch LVIN.

As is well known in the art of digital design, many latches suitable for use as the validation and acceptance latches may have two outputs, Q and NOT (Q), that is, Q and its logical inverse. If such latches are chosen, the one input to the OR gate can, therefore, be tied directly to the NOT (Q) output of the validation latch LVIN. The gate NAND1 and the inverter INV1 can be implemented using well known conventional techniques. Depending on the latch architecture used, however, it may be more efficient to use a latch without an inverting output, and to provide instead the gate NAND1 and the inverter INV1, both of which also can be implemented efficiently in a silicon device. Accordingly, any known arrangement may be used to generate the Q signal and/or its logical inverse.

The data and validation latches LDIN, LDOUT, LVIN and LVOUT, load their respective data inputs when both clock signals (PHO at the input side and PH1 at the output side) and the output from the acceptance latch of the same side are logical "1". Thus, the clock signal (PHO for the input latches LDIN and LVIN) and the output of the respective acceptance latch (in this case, LAIN) are used in a logical AND manner and data is loaded only when they are both logical "1".

In particular applications, such as CMOS implementations of the latches, the logical AND operation that controls the loading (via the illustrated CK or enabling "input") of the latches can be implemented easily in a conventional manner by connecting the respective enabling input signals (for example, PHO and QAIN for the latches LVIN and LDIN), to the gates of MOS transistors connected in series in the input lines of the latches. Consequently, is necessary to provide an actual logic AND gate, which might cause problems of timing due to propagation delay in high-speed applications. The AND gate shown in the figures, therefore, only indicates the logical function to be preformed in generating the enable signals of the various latches.

Thus, the data latch LDIN loads input data only when PHO and QAIN are both "1". It will latch this data when either of these two signals goes to a "0".

Although only one of the clock phase signals PHO and PH1, is used to clock the data and validation latches at the input (and output) side of the pipeline stage, the other clock phase signal is used, directly, to clock the acceptance latch at the same side. In other words, the acceptance latch on either side (input or output) of a pipeline stage is preferably clocked "out of phase" with the data and validation latches on the same side. For example, PH1 is used to clock the acceptance input latch, although PH0 is used in generating the clock signal CK for the data latch LDIN and the validation latch LVIN.

As an example of the operation of a pipeline augmented by the two-wire validation and acceptance circuitry assume that no valid data is initially presented at the input to the circuit, either from a preceding pipeline stage, or from a transmission device. In other words, assume that the validation input signal IN_VALID to the illustrated stage has not gone to a "1" since the system was most recently reset. Assume further that several clock cycles have taken place since the system was last reset and, accordingly, the circuitry has researched a steady-state condition. The validation input signal QVIN from the validation latch LVIN is, therefore, loaded as a "0" during the next positive period of the clock PH0. The input to the acceptance input latch LAIN (via the gate NAND1 or another equivalent gate), is, therefore, loaded as a "1" during the next positive period of the clock signal PH1. In other words, since the data in the data input latch LDIN is not valid, the stage signals that it is ready to accept input data (since it does not hold any data worth saving).

In this example, note that the signal IN_ACCEPT is used to enable the data and validation latches LDIN and LVIN. Since the signal IN_ACCEPT at this time is a "1", these latches effectively work as conventional transparent latches so that whatever data is on the IN_DATA bus simply is loaded into the data latch LDIN as soon as the clock signal PH0 goes to a "1". Of course, this invalid data will also be loaded into the next data latch LDOUT of the following pipeline stage as long as the output QAOUT from its acceptance latch is a "1".

Hence, as long as a data latch does not contain valid data, it accepts or "loads" any data presented to it during the next positive period of its respective clock signal. On the other hand, such invalid data is not loaded in any stage for which the acceptance signal from its corresponding acceptance latch is low (that is, a "0"). Furthermore, the output signal from a validation latch (which forms the validation input signal to the subsequent validation latch) remains a "0" as long as the corresponding IN_VALID (or QVIN) signal to the validation latch is low.

When the input data to a data latch is valid, the validation signal IN_VALID indicates this by rising to a "1". The output of the corresponding validation latch then rises to a "1" on the next rising edge of its respective clock phase signal. For example, the validation input signal QVIN of latch LVIN rises to a "1" when its corresponding IN_VALID signal goes high (that is, rises to a "1") on the next rising edge of the clock phase signal PH0.

Assume now, instead, that the data input latch LDIN contains valid data. If the data output latch LDOUT is ready to accept new data, its acceptance signal QAOUT will be a "1". In this case, during the next positive period of the clock signal PH1, the data latch LDOUT and validation latch LVOUT will be enabled, and the data latch LDOUT will load the data present at its input. This will occur before the next rising edge of the other clock signal PH0, since the clock signals are non-overlapping. At the next rising edge of PH0, the preceding data latch (LDIN) will, therefore, not latch in new input data from the preceding stage until the data output latch LDOUT has safely latched the data transferred from the latch LDIN.

Accordingly, the dame sequence is followed by every adjacent pair of data latches (within a stage or between adjacent stages) that are able to accept data, since they will be operating based on alternate phases of the clock. Any data latch that is not ready to accept new data because it contains valid data that cannot yet be passed, will have an output acceptance signal (the QA output from its acceptance latch LA) that is LOW, and its data latch LDIN or LDOUT will not be loaded. Hence, as long as the acceptance signal (the output from the acceptance latch) of a given stage or side (input or output) of a stage is LOW, its corresponding data latch will not be loaded.

Fig. 4 also shows a reset feature included in a preferred embodiment. In the illustrated example, a reset signal NOTRESET0 is connected to an inverting reset input R (inversion is hereby indicated by a small circle, as is conventional) of the validation output latch LVOUT. As is well known, this means that the validation latch LVOUT will be forced to output a "0" whenever the reset signal NOTRESET0 becomes a "0". One advantage of resetting the latch when the reset signal goes low (becomes a "0") is that a break in transmission will reset the latches. They will then be in their "null" or reset state whenever a valid transmission begins and the reset signal goes HIGH. The rest signal NOTRESTO, therefore, operates as a digital "ON/OFF" switch, such that it must be at a HIGH value in order to activate the pipeline.

Note that it is not necessary to reset all of the latches that hold valid data in the pipeline. As depicted in Fig. 4, the validation input latch LVIN is not directly reset by the reset signal NOTRESETO, but rather is reset indirectly. Assume t hat t he r eset s ignal NOTRESETO drops t o a "0". T he v alidation output signal QVOUT also drops to a "0", r egardless of its previous s tate, whereupon the input to the acceptance output latch LAOUT (via the gate NAND1) goes HIGH. The acceptance output signal QAOUT also rises to a "l". This QAOUT value of "1" is then transferred as a "1" to the input of the acceptance input latch LAIN regardless of the state of the validation input signal QVIN. The acceptance input signal QAIN then rises to a "1" at the next rising edge of the clock signal PH1. Assuming that the validation signal IN_VALID has been correctly reset to a "0", then upon the subsequent rising edge of the clock signal PH0, the output from the validation latch LVIN will become a "0", as it would have done if it had been reset directly.

As this example illustrates, it is only necessary to reset the validation latch in only one side of each stage (including the final stage) in order to reset all validation latches. In fact, in many applications, it will not be necessary to reset every o ther validation l atch: If the r eset signal N OTRESET0 can be guaranteed to be low during more than one complete cycle of both phases PH0, PH1 of the clock, then the "automatic reset" (a backwards propagation of the rest signal) will occur for validation latches in preceding pipeline stages. Indeed, if the reset signal is held low for at least as many full cycles of both phases of the clock as there are pipeline stages, it will only be